Non-context-free languages

Reading

Section 2.3

Practice problems (page 128): 2.2, 2.30, 2.31, 2.32, 2.38

The pumping lemma for context-free languages

The pumping lemma for context-free languages is a bit more complicated:

> **Pumping Lemma for CFLs**
>
> If A is a context-free language, then there is a number p, so that any string $s \in A$ of length at least p may be divided into 5 pieces $s = uvxyz$ so that:
>
> 1. $uv^ixy^iz \in A$ for all $i \geq 0$,
> 2. $|vy| \geq 0$,
> 3. $|vxy| \leq p$.

So instead of having a single piece that can be pumped, it has two pieces that are "pumped together". Condition 2 says that at least one of the strings v, y is non-empty.

Proof of the pumping lemma

The proof involves thinking about the parse tree for a string. For such a parse tree, we want to consider paths from the head (start nonterminal) to the different "leaves" (terminals). The *length* of such a path is the number of "edges" (so a path of length 3 will contain a total of 4 symbols along the way).

The following result is obtained by using a standard counting argument:

> If the depth of the parse tree (the largest length of a path in the tree) is p, and the largest number of items that can occur in the right hand side of a production rule in the grammar is b, then the generated string has at most b^p terminals.
>
> Conversely, if a string has length more than b^p, then there must be a path in the parse tree of length more than p.

This follows somewhat simply: The production rules tell us how many children each node can have. so the start node can have at most b child nodes. Each of these can have at most b child nodes, for a total of b^2 nodes possible at depth 2. Continuing this logic, there are at most b^k nodes at depth k.

Now we proceed with the proof of the pumping lemma. For the rest, we denote by b the largest number of symbols used on the right hand side of a production rule, and by q the number of nonterminals. Then consider $p = b^{q+1}$. This will be the pumping length.

Suppose now that we have a string of length at least $p > b^q + 1$. By our previous observation, there must be some path that has length at least $q + 1$. This means that it contains at least $q + 2$ elements in total, so at least $q + 1$ elements if we exclude the leaf element.

Now all these elements along the way must be non-terminals, as they are the only ones that can appear in a non-leaf position. So we have at least $q + 1$ non-terminals down that path. Since there are only q distinct non-terminals in the language, it must be the case that two elements down that list are the same nonterminal, R.

Now we think of the derivations that correspond to this parse tree. We must have first a derivation

$$S \Rightarrow^* uRz$$

where we've expanded all other derivation steps except the one with top node the first occurence of R (and those below it). Now if we expand the derivation that has at its top this R, it produces another R somewhere inside it, so there is some derivation:

$$R \Rightarrow^* vRy$$

Finally, the inner R also has a certain derivation that results in a string x:

$$R \Rightarrow^* x$$

The normal derivation of the string s we started would then be the combination of these:

$$S \Rightarrow^* uRz \Rightarrow^* uvRyz \Rightarrow^* uvxyz$$

so $s = uvxyz$.

We can see now how we could pump such a string: By omitting or repeating the step

$$R \Rightarrow^* vRy$$

any number of times. This is the source of the pumping, that R can produce itself with some extras on either side. The resulting strings are exactly of the form uv^ixy^iz.

What is left is to determine the other conditions in the lemma, regarding the lengths of vy and vxy. For that we need to have made some choices along the way carefully:

- First, we choose the parse tree to be the smallest possible out of all parse trees that would have derived that string in our grammar.
- Second, we choose the longest path in the tree instead of any long enough path.
- Third, we must choose our R carefully. In particular, we need to choose it amongst the bottom $q + 2$ elements in the path (we know that's already too many elements to go by without duplication).

The second and third conditions together guarantee that the parse tree that sits below the chosen "outer" R has depth at most $q+1$, and hence has in total at most $b^{q+1} = p$ terminals at the resulting derived string. Since the derived string is vxy, that string has length at most p, proving the third condition in the pumping lemma.

Lastly, we need to consider the second condition in the pumping lemma. If the length of vy was 0, i.e. if both v and y were empty, then the derivation from the "outer" R to the "inner" R would end up being $R \Rightarrow^* R$ (e.g. other non-terminals in the rules along the way became ϵs). But in that case we could replace the tree for the outer R with the tree for the inner R, and obtain an overall smaller tree that still derives the string s. This contradicts our first assumption above. Therefore the length of vy must be non-zero, and this completes the proof of the pumping lemma.

Examples of non-context-free languages

We can use the pumping lemma to conclude that a number of languages are not context-free.

> The language $B = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

This should now be a familiar pattern. Suppose that B was context-free, and hence there is a CFG generating it. Let p be the pumping length, and consider the string $s = a^p b^p c^p \in B$. We will show that it cannot be pumped.

The pumping lemma guarantees that we can write $s = uvxyz$ where $|vxy| \leq p$. Therefore this part cannot contain both as and cs, as it is not long enough to cross over the bs. It may also consist of only bs.

This means that $uv^2xy^2z \in B$ has either have the same number of as or the same number of cs, while one of the other symbols' count goes up. This is a contradiction.

Let us look at another example:

> The language $C = \left\{a^i b^j c^k \mid 0 \leq i \leq j \leq k\right\}$ is not context-free.

The idea is similar, we look at the string $s = a^p b^p c^p$ as above, where p is the pumping length. As before, vxy must either contain no cs or contain no as.

- If it contains no cs, then $uv^2xy^2z \in C$ has more than p copies of either a or b, which contradicts the $i, j \leq k$ assumption.
- If it contains no as, then we look at $uxz \in C$, which has less than p copies of either b or c, which contradicts the $i \leq j, k$ assumption.

Lastly, consider the following:

> The language $D = \{ww \mid w \in \{0,1\}^*\}$ is not context-free.
>
> Constrast this with the language of palindromes: $\left\{ww^R \mid w \in \{0,1\}^*\right\}$, which is context-free.

It is instructive to think first for a second where the pumping lemma would break down for palindromes. In fact the palindrome grammar is simple:

```
S -> aSa | bSb | cSc | ... | eps
```

The nonterminal R in the proof of the pumping lemma would be S, and it always appears symmetrically in the middle. Repeating or removing that middle part does not break the pattern.

Now let us show that our language D above is not context-free. If it were, and if p was its pumping length, then consider the string $s = 0^p 1^p 0^p 1^p$ and how it would break into parts $s = uvxyz$. No matter how we arrange it, the strings v, y will contain 0s from one of the sections but not the other, or 1s from one of the sections but not the other. In either case, replacing vxy with just x results in a string of the form $0^k q^l 0^s q^m$ where either $k \neq s$ or $l \neq m$. It is certainly not a repetition of a string.